PROLOG语言及其在土木工程中的应用

郭秀玲　管　齐　编著

冶金工业出版社

兼容机，因此有推广的普遍意义。本书得到全国土木工程学会计算机应用委员会的大力支持，在此特表示感谢！

编者

1988年1月

内 容 简 介

本书是结合有关PROLOG语言的专著并联系土木工程实际应用而编写的。

书中首先对PROLOG作了深入浅出的介绍，并致力于理论与土木工程实际应用的密切结合，力求内容丰富、简练、通俗易懂。

全书共分七章，包括基本概念、语法规则、常用内部谓词、土木工程应用实例和A.D.A PROLOG版本简介。在土木工程应用实例中结合工程实际问题介绍PROLOG编程方法以及在研制专家系统中的应用，内容新颖，对我国土木工程界推广、应用PROLOG语言和开发相应智能专家系统具有参考价值。

本书可供研究生、大学生、科技人员、管理人员，尤其是非计算机专业人员参考，也可供各类高等文、理科院校选为研究生教材。

PROLOG语言及其
在土木工程中的应用

郭秀玲　管　齐　编著

*

冶金工业出版社出版发行
（北京北河沿大街嵩祝院北巷39号）
新华书店总店科技发行所经销
冶金工业出版社印刷厂印刷

*

787×1092 1/32　印张 5 1/4　字数 114 千字
1989 年 7 月第一版　1989 年 7 月第一次印刷
印数00,001～30,000册
ISBN 7-5024-0456-2
TU·23　定价1.80元

序

PROLOG是建立在符号逻辑基础上的一种新型的程序语言，通常也称之为"逻辑编程语言"。近年来，由于PROLOG是开发专家系统的一个理想语言，所以在日本已把它作为第五代计算机系统的核心语言，因而引起广泛的重视。

逻辑编程语言PROLOG以一阶谓词和关系理论为基础，采用接近自然语言逻辑描述形式来告诉计算机所应完成的任务，并根据数据库中的知识自动进行推理。PROLOG这种自动进行逻辑推理的功能，远远超过了FORTRAN等高级语言，因此PROLOG的特点是比这些语言简单，编写程序容易，可以很快写出清楚易读，简短而很少有错误的程序。

对我国来说，面临先进国家研制第五代计算机和智能的挑战。在制定新的技术革命对策时，又增加了一个需要考虑的重要方面。在人工智能应用程序的设计中和专家系统的研制开发中，都将以PROLOG作为重要的工具。因此，中国土木工程学会计算机应用学会组织出版《PROLOG语言及其在土木工程中的应用》一书是及时的，我们相信这是广大科技工作者所欢迎的。

上海交通大学土木系郭秀玲、管齐等一直活跃在计算机在土木工程中应用的第一线。他们在运用PROLOG语言方面又领先了一步。我们非常感谢他们毫无保留地将他们许多宝贵经验告诉大家，使我们在应用PROLOG语言方面可以有一条捷径。我们相信，这本书的出版，加上在计算机上的实践，人们是一定能够掌握PROLOG的。

赵超樊

1988年8月

前　言

PROLOG是一种新颖的建立在符号逻辑基础上、简单而功能却很强的逻辑程序设计语言。它是“Programming in Logic”的缩写。由法国马赛大学的Colmerauer和Roussel于1972年设计和实现的。它具有递归、回溯、模式匹配、数据库等很强的功能。它以一阶谓词及关系理论为基础，使用中接近自然语言逻辑描述形式来告诉计算机所应完成的任务，并根据存放在数据库中的知识，自动进行推理。PROLOG这种自动逻辑推理功能，远远超过FORTRAN等高级语言，其特点是比这些语言简单、易学、易掌握，可以很快写出清楚、简短而且很少有错误的程序。

PROLOG（更一般叫做“逻辑程序设计”）的出现被认为是计算机科学中一个极重要的进展。它适用于非数值处理、自然语言的理解、数学论证、数据库系统，专家系统等人工智能领域。近年来，由于PROLOG是开发专家系统的一个理想语言，日本已把它作为第五代计算机的核心语言，更引起广泛的重视。七十年代在欧洲传播，近年来美国十分重视，目前正在全世界范围内迅速传播。

我国计算机界也对此日益关注且很多人开始从事这方面的研究。由于该语言兴起较晚，我们结合土木工程的应用编著这本书目的是为了普及推广PROLOG语言及其在各项工程中的应用。

本书所介绍的A.D.A Prolog版本适用于IBM PC及其

目　录

第一章 基本概念

PROLOG是一种计算机程序设计语言。同其他程序设计语言一样，PROLOG也存在着不同的版本(可参见附录B)。由于本书采用A.D.A Prolog，其参考手册强调必须参考英国爱丁堡大学的W.F.克劳克辛(Clocksin)和C.S.梅里雪(Mellish)的经典著作《PROLOG编程 (Programming in Prolog)》，所以在本书中，我们以该书中的“核心PROLOG”为蓝本，参照其语法和语义进行介绍，但所有的例子都在A.D.A Prolog系统上调试通过。

虽然PROLOG有不同的处理系统，但都是用于解决有关目标及目标之间的关系问题。本章的目的，仅通过程序介绍该语言的基本要素，以及必须弄清的一些基本概念，如事实、询问、变量、连接和规则。

一个问题若可表示成若干对象和它们关系的形式，并用计算机解这类问题时，就可用PROLOG语言。例如：我们说“约翰喜欢足球”，用PROLOG可以将它表示成事实：

```
likes (john, football).
```

它表明对象“约翰”和另一个对象“足球”之间有一个关系“喜欢”(likes)。这种关系是按一定的次序来表示的，“喜欢”者放在前面，被“喜欢”者放在后面。这样放置是为了符合英语的习惯，但对PROLOG而言，将其次序颠倒并不会引起出错。再如：“pdp11是计算机”，可表示成如下事实：

```
computer(pdp11).
```

它有一个对象“pdp11”，一个关系即“是计算机”。这就是

说，并不是所有的关系都要涉及问题所包括的所有对象，而关系完全依赖于问题的本身。

通常我们习惯于用一些规则去描述对象之间的关系，例如:“如果是约翰喜欢的汤姆也喜欢”，用PROLOG表示成规则:

```
likes(tom, X):-likes(john, X).
```

“如果X是计算机，则X能做加法”，表示成规则:

```
can__do__addition(X):-computer(X).
```

规则往往是简单的，但它作为定义却能为人们所接受，注意到这一点是很重要的。毕竟我们不能对某件事的所有有关情况给予定义，但在解决一个特定问题时，只需注意对解决问题有用的规则。因此，我们应该认真设计一个能满足解决问题的理想而又简化的定义。

假设我们现给出事实,“iBM-PC是计算机”及有关规则:

```
computer(iBM__PC).
can_do_addition (X):-computer(X).
```

就构成了一个PROLOG程序。对此程序提问:“iBM-PC能做加法吗？”，则PROLOG表示为如下形式:

```
root/user/?-can__do__addition(iBM__PC).
```

执行回答的结果是:

```
Yes.
```

这里，PROLOG将搜索我们所给有关“iBM-PC”的事实，然后根据提问回答“Yes”或“No”。

由上例可见，一个PROLOG程序由以下三部分组成:

（1）描述客观事物的一些对象及其关系的事实。

（2）定义有关一些对象和关系的规则。

（3）对有关对象及其相互关系的询问。

所以，我们可以把PROLOG看作为一个事实与规则的库，它将根据事实和规则回答询问。编写一个PROLOG程序，应包括提供所有的事实和规则。PROLOG系统将使计算机作为一个事实和规则的库来使用，其程序语句的执行，是按严格的次序从上至下，从左到右（深度优先），以模式匹配为基础完成的，即提供了从一个事实推出另一个事实的方法。

同时，PROLOG也是一种会话式语言，即具有很好的人机对话的功能。当通过键盘输入要解决问题所需的事实和规则后，由用户提问，PROLOG就给出回答并输出结果。

一个PROLOG程序是由事实，询问及规则这三大要素组成的。

一、事　实

事实是由关系和对象构成的。关系和对象可由编程者自己定义。因此，关系和对象的名字是任选的。

一个事实是一条简单句，也就是一条语句。它必须有一个关系名，说明对象与对象之间的关系。例如："汽车有轮子。"这个句子是一个事实，它包含两个对象"汽车"和"轮子"和一个关系"有"，在PROLOG中，这个事实可用如下标准形式表示：

```
has(car, wheels).
```

在此语句中，要求：

（1）对象和关系名必须以小写字母开头，如：has，car，wheels。

（2）关系写在前面（在圆括号外），所有对象写在圆括号内，彼此用逗号隔开。

（3）每个事实的结尾必须写上"."

这个语句是一个事实，“有”在这句子中是谓词。所以，关系名又叫谓词。当用事实来定义对象之间的关系时，要注意圆括号中对象的书写顺序。实际上，这个次序是任意的，但一经规定某顺序后，就要保持它的一致性。如上面的例子，has (car wheels). 也可以写为has(wheels, car). 但两者的意思不同，前者是说“汽车有轮子”，而后者是说“轮子有汽车”。这两种不同的意思，对PROLOG来说，两者都是可接受的。这里句子中的“有”，“汽车”和“轮子”都是PROLOG的原子，而其中“汽车”和“轮子”又是常数。

下面给出一些事实的例子：

owns(susan, horse).	“susan owns horse”
female(jane).	“jane is female”
likes(jill,meat).	“jill likes meat”
valuable(gold).	“gold is valuable”

各事实的中文意思分别为：

“苏珊有一匹马。”

“詹妮是女性。”

“吉尔喜欢肉。”

“黄金是贵重的。”

以上每句中所含有的名字，它是一个特殊的独立对象，当使用一个名字时，首先区分不同的解释，以便确定它的含义。一旦编程者确定名字的含义，只要遵守决定，编程中就不会有问题。

通常我们将每一个事实中圆括号里的对象的名字称为“参量”，而位于括号前的关系名字称为“谓词”。所以，female和valuable都是只有一个参量的谓词，又叫一元关系。而owns和likes是有两个参量的谓词，又叫二元关系。由于

对象和关系的名字是任选的，我们完全可以用k(j,m)，代替likes(jill, meat).

在 PROLOG中一个关系可有多个参量。如："贝尔给玛丽书"，可表示为：

gives(bill, mary, book).

这里gives是拥有bill，mary和book三个参量的谓词，又叫三元关系。通过谓词使三个孤立互不相干的对象之间产生了联系。所以，PROLOG中事实允许编程者表达任意对象中的关系。

当我们为解决某一特定问题而给出一系列事实（加上规则）时，在PROLOG中这些事实的集合被称为"数据库"。

二、询　　问

一旦已知一些事实后，我们就可以对一些事实询问。询问的目的是为了求解问题，找出问题的答案。例如有如下事实的数据库：

has(car, wheels).

has(car, frame).

has(car, windshield).

has(car, engine).

将这些事实输入PROLOG系统后，就可以进行询问。PROLOG中的询问很象一个事实，但它只能在系统提示符?-（A.D.A Prolog的提示符是root/user/?-）出现后才能进行询问，而事实则是通过其他方法存放到系统中去的。例如：

?-has(car,wheels).

相当于问"汽车有轮子吗？"或者"汽车有轮子是事实吗？"，一旦对PROLOG提出询问，它将搜索事先已输入的事实组成

的数据库，寻找与询问中的事实匹配的那些事实。两个事实匹配的意思是指它们的谓词相同，且对应的参量也相同。如果找到匹配的事实，则PROLOG回答“Yes”，如果数据库中不存在这样的事实，就回答“No”。在终端屏幕上，回答将显示于所询问问题的下一行位置上。如果将上述的数据库调入PROLOG系统，并进行以下的询问，就可以产生结果。

```
root/user/?-has(car,wheels).
Yes.
root/user/?-has(frame, car).
No.
root/user/?-likes(mary, book).
No.
root/user/?-has(car, engine).
Yes.
```

在上述询问的回答中，第三个询问的回答“No”是因为数据库中无“玛丽喜欢书”的事实。所以，PROLOG回答“No”通常是指“不知道”。

如果我们要询问PROLOG这样一个问题，“汽车有什么部件？”为回答这样的询问，就引出了“变量”这个概念。

三、变　量

假设有数据库：

```
has(car, wheels).
has(car, frame).
has(car, windshield).
has(car, engine).
```

PROLOG中有一种直接询问的方法，即：

?-has(car, Part).

它的含义是“汽车有什么部件？”。这时，我们并不知道Part代表的是什么对象，等待PROLOG给出可能的回答。这就是说，在PROLOG中不仅可以定义特定的对象名字，还可以定义待定的对象名字Part，或者可以使用X这样的名字来代替待定的对象名字Part，我们把这种名字称为变量。PROLOG使用变量时，该变量可以是已定义的，也可以是未定义的。当变量已定义，即该变量已代表了一个对象。反之，变量未定义，即尚不知该变量代表什么对象。变量必须以大写字母开头，PROLOG就是依此来区别变量和一个特定对象的名字的。上例中的Part中的大写字首“P”说明Part是变量。

当向PROLOG询问一个带有变量的问题后，PROLOG就搜索数据库中所有的事实，以便寻找代表变量的对象。这里，PROLOG将使Part等于任何要求的常数（对象），以使询问与数据库中的一个事实相匹配，故PROLOG将回答：

Part=wheels.

找到一个答案后，便自动又立即显示More?(Y/N)：意思为还要找一个吗？让你选择Y(Yes) 或n(No) 键，若不需要则按n键，则停。若按y键，则下一个回答将出现：

Part=frame.

又显示：

More? (Y/N):y

继续下去，PROLOG将给出回答，Part=windshield，和Part=engine，最后，你将看到：

More?(Y/N):y

No.

说明数据库已搜索完毕。

从上述过程来看，PROLOG回答此问题的过程是这样的：PROLOG接受询问时，变量是无值的。它要在数据库中寻找与它匹配的事实。由于这个未定义的变量作为参量出现，PROLOG便将用事实中处于同样位置的任何其它参量来匹配。此例中，PROLOG搜索数据库寻找以“has”为谓词，且有二元关系，其第一个参量为“car”的任何事实，找到这样一个事实后，变量Part就和第二个参量相匹配。由于PROLOG是按输入数据库的顺序查找的，所以，has(car, wheels). 先找到，这时Part代表“wheels”,或者说Part被赋值为“wheels”,同时PROLOG将记下匹配事实的位置。PROLOG一次只找出一个与询问匹配的事实，并将变量的值输出。然后，PROLOG停下等待进一步的指令。若我们得到一个回答后满足了，则按N或n键，PROLOG停止搜索，但若还想要更多的回答，则按Y或y键，PROLOG将从数据库的位置指针处继续向下寻找，再次回答询问。这就意味着要用另一个答案来满足此询问，找出Part可能代表的另一个对象，那么PROLOG将丢弃Part的 wheels意义，重新把它当作无值的继续搜索，下一个能匹配的事实是 has(car, frame). 即 Part代表 frame. 如果再打入一个Y或y 键，PROLOG 继续寻找，直到此后再无匹配事实，则PROLOG打印出 “No”，并停止搜索，这时，允许再询问或输入新事实。

四、连　　接

假设有如下数据库：

```
has(car, wheels).
has(car, frame).
```

has(bicycle, frame).

has(wheels, hubs).

has(bicycle, lighting__system).

如果我们问“汽车有轮子并且轮子有轴吗？”这是一个涉及到更多更复杂的关系问题。即该问题包括两个要满足的独立目标，“汽车有轮子吗？”及“轮子有轴吗？”。由于PROLOG编程者经常要用到这种组合型问题，所以，给它一个特殊的表示法。

对“has car wheels and has wheels hubs?”

这里的and—“并且”表示把两个有关的询问“连接”起来，并要求二者同时得到满足。在PROLOG中，我们在两个目标中采用一个“，”号来表示：

root/user/?-has(car, wheels), has(wheels, hubs).

“，”代表“并且”，它把为回答问题所必须满足的任意个数的不同目标分隔开。当对PROLOG询问一系列用“，”隔开的目标，PROLOG便搜索数据库，依次为每一目标寻找匹配的事实。当所有目标都满足时，才回答“Yes”。在上述例子中，因为数据库中有“汽车有轮子”和“轮子有轴”的事实，故两个目标均得到满足，所以，整个问题的回答是“Yes”。即：

root/user/?-has(car, wheels), has (wheels, hubs).

Yes.

如果又问：

root/user/?-has(car,wheels), has(car, chain).

No.

对数据库搜索，其中第一个目标回答是“Yes”，而第二个目标，由于数据库中没有has(car , chain). 这一事实，所以第二个目标得不到满足，整个询问的回答是“No”。此外，连接和变量可以结合使用，如我们可以询问“汽车和自行车有什么相同的部件？”这个询问也含有两个目标：首先找出汽车的组成部件，然后找出自行车是否也由此部件组成。在PROLOG中，这两个目标连接可写成如下形式：

root/user/?-has(car, X), has(bicycle, X).

为了回答上述询问，PROLOG先回答第一个目标，如果能满足第一个目标，然后再去看是否满足第二个目标，如果第二个目标也满足，PROLOG将在数据库中记下目标的位置，同时得到满足两个目标询问的回答。

应当注意，每一个目标一定有其自己的位置指针。如果第二个目标得不到满足，PROLOG便重新满足它的前一个目标即指针继续往下找第一个目标的答案。记住，每个目标进行第二次查找都是从其各自的位置指针开始，而不是从数据库的头开始。对于“汽车和自行车有什么相同部件？”这种组合型问题的询问，PROLOG回答的步骤如下：

（1）为满足第一个目标搜索数据库，查找第一个事实。由于第二个自变量X未定义，它可与任何对象匹配。在上述数据库中，最先可匹配的事实是has(car, wheels). 这时问题中所有X都被定义为“wheels”. PROLOG记下数据库这个位置，以便在需要重新满足该目标时用。从此后的X就代表“wheels”，直到下次需要重新匹配时才能改变。

（2）因为第二个目标是has(bicycle, X)，此时X即“wheels”，在数据库中搜索has (bicycle, wheels)，可以看到数据库中没有此事实，该目标没被满足。当一个目标没被满